REMARKS

Claims 1-30 are pending in the above-identified application. Claims 1, 15 and 27 are independent.

The examiner indicated that China Patent 1138175 which was listed in the March 2, 2004 IDS failed to comply with 37 C.F.R. §1.98(a)(2) in that a copy of the patent was not included. Submitted with this Amendment is a Supplemental IDS listing the International Search Report (ISR) prepared in relation to the PCT application corresponding to the above-identified application, and U.S. Patent No. 5,850,551 to Takayama. As indicated in the ISR, the Takayama reference is the U.S. counterpart to China Patent 1138175.

The examiner objected to claims 10-12 on the ground that they recited "memory back" instead of "memory bank." Applicant thus amended claims 10-12 to correct the typographical error and recite "memory bank." Applicant thanks the examiner for pointing out this typographical error.

The examiner rejected claims 27-30 under 35 U.S.C. §101 on the ground that "[t]he computer program product of claims 27-30 is embodied in an information carrier, which could mean non-statutory electronic signals."

Applicant amended independent claim 27 to recite "[a] computer program product, for vectorizing memory access instructions, the computer program product residing on a machine-readable medium for storing computer instructions that, when executed, cause data processing apparatus to...." Applicant's computer program product resides on a machine-readable medium that store instructions and is therefore a tangible product that does not include non-statutory electronic signals. Applicant thus traverses the examiner's §101-based rejections.

Applicant also amended claims 27, as well as its dependent claims 28-30 for greater clarity.

The examiner rejected claims 1-6, 11-18 and 23-30 under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 6,356,260 to Pentkovski et al. The examiner further rejected claims 7-10 and 19-22 under 35 U.S.C. 103(a) as being unpatentable over Pentkovski in view of the Microsoft Press Computer Dictionary.

Specifically, regarding independent claim 1, the examiner stated:

> 3. With respect to claim 1, Pentkovski et al. teach a method comprising:
>
> converting memory access instructions in a source code into a standard format, in fig. 5, where store or load commands that address consecutive locations in memory are first put in order in intermediate buffer 362 and combined in the write combining buffers 470.
>
> generating partitions containing formatted memory access instructions, in fig. 5, reference characters WO, W1, W2 and W3.
>
> generating a match set, the match set including matches of instruction patterns to the formatted memory access instructions in the partitions, in fig. 5, intermediate buffer 362, where, for example, store EO, E1, E2 and E3 form a match set.
>
> transforming the matches to vector memory access instructions, in fig. 5, reference numeral 650, where store EO, E1, E2 and E3 are combined into one memory access instruction WO. See the related discussion in col. 6, lines 38-54.

Applicant respectfully disagrees. Applicant amended independent claim 1 to clarify that the memory access instruction are converted into standard formatted memory access instructions. Support for this clarification is found throughout the application including, for example, at page 3, lines 21-31 of the originally filed application.

Applicant's amended independent claim 1 thus recites "converting memory access instructions in a source code into standard formatted memory access instructions; generating partitions containing the standard formatted memory access instructions; generating a match set, the match set including matches of instruction patterns to the standard formatted memory access instructions in the partitions; and transforming the matches to vector memory access instructions."

Applicant's method converts two or more memory access instructions into a single vectorized memory access instruction. As explained in the originally filed application:

> **Vectorizing two or more memory reads (or two or more memory writes) does not require that the read (or write) addresses be continuous. For example, a first SRAM read instruction 154 that would read 4 bytes starting at byte Addr+0 and a second SRAM read instruction 160 reads 4 bytes starting at byte Addr+8 can be vectorized into a single vectorized SRAM read instruction 162 that would read 12 bytes starting at byte Addr+0 and including byte Addr+11. Although 4 bytes, Addr+4, Addr+5,**

> **Addr+6 and Addr+7 are not used, it still saves time to perform one memory read to fetch the 12 bytes rather than two memory reads to fetch 4 bytes each. The memory read to four unused bytes Addr+4, Addr+5, Addr+6, Addr+7 are referred to as an instantiable memory read. (page 5, line 25, to page 6, line 5, of the originally filed application)**

In contrast, Pentkovski describes a method and apparatus for efficient utilization of write-combining buffers through the implementation of intermediate buffers that facilitate a quick transfer of data to the write-combining buffers (col. 1, lines 6-9). Particularly, as Pentkovski explains:

> **The sequence 410 includes a number of non-temporal (NT) stores 420.sub.1-N, that are intermixed with loads and other instructions 422.sub.1-M. In the example of FIG. 4, N=8 and M=8. The intermediate stores 460 transform each non-adjacent NT store 420.sub.1-N to a cacheable store and an adjacent NT store, as described in detail in the following sections.**
>
> **The WC buffer unit 470 includes four fill buffers: buffers W0, W1, W2, and W3. In one embodiment, each buffer W0, W1, W2 and W3 has four storage locations each being 8 bytes wide. In alternate embodiments, a greater or lesser number of storage locations may be implemented. In addition, in alternate embodiments, each storage location may accommodate fewer or a greater number of bytes. In one embodiment, the four fill buffers W0, W1, W2, and W3 are located in the L1 cache controller buffers 370. As shown in FIG. 4, the buffers W0-3 are partially filled, where a "-" represents non-valid data, an "x" represents valid data, and A0, A1, B0, B1, B2, C0, C1, and D0 represent valid data.**
>
> **The intermediate stores 460 write non-adjacent data A0, C0, D0, B0, A1, C1, B1, and B2, respectively, to the buffers W0, W1, W2, and W3. In the example shown in FIG. 4, A0 and A1 are written to the buffer W0; B0, B1, and B2 are written to the buffer W1; C0 and C1 are written to the buffer W2; and D0 is written to the buffer W3. (col. 5, line 59, to col. 6, line 16)**

Pentkovski further describes that non-adjacent stores in the code are gathered in an intermediate buffer in such a manner that non-temporal data corresponding to adjacent buffer locations are grouped together (col. 6, lines 21-37). Thus, unlike applicant's independent claim 1, Pentkovski does not convert memory access instructions in a source code into instructions having a standard format, but rather arranges the data corresponding to the non-temporal stores in an intermediate buffer according to the adjacency of the memory locations corresponding to those non-temporal stores. As shown in the illustration of the intermediate buffer 263 in FIG. 5,

and as explained by Pentkovski, "[t]he code sequence 520 gathers the non-adjacent stored data in an intermediate, software-allocated buffer in cacheable memory such as the intermediate buffer 362. The intermediate buffer contains E0-E3, F0-F3, G0-G3, H0-H3" (emphasis added, col. 6, lines 21-25). Accordingly, Pentkovski does not disclose or suggest at least "converting memory access instructions in a source code into standard formatted memory access instructions," as required by applicant's independent claim 1.

Further, although Pentkovski, as noted, performs a partitioning operation to arrange data of non-temporal stores according to the memory location adjacency of that data, Pentkovski does not describe that the non-temporal instructions themselves are partitioned, nor does Pentkovski describe partitioning of any type of instruction. Thus, Pentkovski does not disclose or suggest at least "generating partitions containing the standard formatted memory access instructions," as required by applicant's independent claim 1.

Moreover, Pentkovski also does not describe anywhere that any type of pattern matching operations are performed, let alone instruction pattern matching operations. Instruction pattern matching would be unnecessary for Pentkovski's since what Pentkovski stores in the intermediate buffer (and subsequently in the write-combine buffers) is data and not instructions. Therefore, Pentkovski also does not disclose or suggest at least "generating a match set, the match set including matches of instruction patterns to the formatted memory access instructions in the partitions," as required by applicant's independent claim 1.

While Pentkovski suggest that data is arranged into groups in the intermediate buffer based on the locations of the stores to enable data transfer to be performed more quickly, Pentkovski does not describe that instructions are vectorized such that a single memory access instruction replaces two or more source instructions used in the source code. Indeed, as noted, the intermediate buffer contains data and not instructions, and in any event, Pentkovski makes no mentions of generating resultant vectorized instructions. Thus, Pentkovski does not disclose or suggest "transforming the matches to vector memory access instructions," as required by independent claim 1.

Since Pentkovski does not disclose or suggest any of "converting memory access instructions in a source code into standard formatted memory access instructions," "generating partitions containing the standard formatted memory access instructions," "generating a match set, the match set including matches of instruction patterns to the formatted memory access instructions in the partitions" and "transforming the matches to vector memory access instructions," applicant's independent claim 1 is therefore patentable over the cited art.

Claims 2-14 depend from independent claim 1 and are therefore patentable for at least the same reasons as independent claim 1.

Independent claims 15 recites "converting memory access instructions that read or write less than a minimum data access unit (MDAU) to memory access instructions that read or write a multiple of the minimum data access unit; converting the memory access instructions into a format including a base address plus an offset; grouping subsets of the converted memory access instructions into partitions; and vectorizing the converted memory access instructions in the subsets that match instruction patterns." For at least similar reasons as those provided with respect to independent claim 1, none of the above features of independent claim 15 is disclosed by the cited art. Accordingly, independent claim 15 is patentable over the cited art.

Claims 16-26 depend from independent claim 15 and are therefore patentable for at least the same reasons as independent claim 15.

Independent claim 27 recites "convert memory access instructions residing in a source code into standard formatted memory access instructions; generate partitions containing the standard formatted memory access instructions; generate a match set, the match set including matches of instruction patterns to the formatted memory access instructions in the subsets; and transform the matches to vector memory access instructions." For at least similar reasons as those provided with respect to independent claim 1, none of the above features of independent claim 27 is disclosed by the cited art. Accordingly, independent claim 27 is patentable over the cited art.

Claims 28-30 depend from independent claim 27 and are therefore patentable for at least the same reasons as independent claim 27.

In addition, applicant contends that many of the features recited in the other claims further distinguish the invention over the cited art

For example, as noted above, the examiner rejected claim 5 under 35 U.S.C. §102(b) as being anticipated by Pentkovski, stating:

> 7. With respect to claim 5, Pentkovski et al. teach the method of claim 4 in which applying comprises limiting a subnode of a partition to memory access instructions directed to a specific memory bank, in fig. 5, where the memory access instructions directed towards the E memory bank are put in the block WO. (Office Action, page 5)

Claim 5 recites "[t]he method of claim 4 in which applying comprises limiting a subnode of a partition to memory access instructions directed to a specific memory bank." As explained in the originally filed application:

> **A memory access partition contains groups of memory access instructions inside one basic block. All memory access instructions inside a group (also called subnode of a memory access partition) perform the same memory access (write or read) to the same memory bank. The compiler 18 vectorizes (108) multiple memory access instructions based on instruction pattern matching performed for each group (subnode) of each memory access partition.**
>
> **As shown in FIG. 3, a set of rules are used to generate (106) memory access partitions, such as memory access partitions 200, 202, 204. A data flow graph 206 is generated. In this example, the data flow graph 206 includes three basic blocks 208, 210, 212. Each of the basic blocks 208, 210, 212 include memory access instructions.**
>
> **For each basic block 208, 210, 212, in the data flow graph 206, the following principles are applied to generate the corresponding memory access partition 200, 202, 204, respectively. Each subnode of a partition contains only memory access operations to a specific memory bank. For example, subnode 214 includes reads to a SRAM memory bank. Subnode 216 includes a read to a DRAM memory bank. (page 4, lines 4-24 of the originally filed application).**

Thus, the partitions include subnodes, and each subnode includes one type of memory access instruction with respect to one memory bank.

While, as noted above, Pentkovski's intermediary buffer 362 is arranged so that its various parts correspond to different write-combining buffers, nowhere does Pentkovski describe that such parts are divided into subgroups (or subnodes). Further, as also noted above,

Pentkovski's intermediate buffer stores data and not memory access instructions. It follows that Pentkovski does not describe that the intermediary buffer includes partitions that include subnodes that hold memory access instructions. Thus, Pentkovski does not disclose or suggest at least "limiting a subnode of a partition to memory access instructions directed to a specific memory bank," as required by applicant's claim 5. Claim 5 is therefore patentable over the cited art.

Claims 6-12 depend from claim 5 and are therefore patentable for at least the same reasons as claim 5.

Claim 16 recites "generating subnodes in partitions, the subnodes including memory access instructions directed to a memory bank and performing the same operation." For at least the same reasons as those provided with respect to claim 5, at least this feature is not disclosed by the cited art. Accordingly, claim 16 is patentable over the cited art.

Claims 17-24 depend from claim 16 and are therefore patentable for at least the same reasons as claim 16.

Claim 30 recites "generate subnodes in partitions, the subnodes including memory access instructions directed to a memory bank and performing the same operation." For at least the same reasons as those provided with respect to claim 5, at least this feature is not disclosed by the cited art. Accordingly, claim 30 is patentable over the cited art.

In addition, and as noted above, the examiner rejected claim 13 under 35 U.S.C. §102(b) as being anticipated by Pentkovski, stating:

> 11. With respect to claim 13, Pentkovski et al. teach the method of claim 1 in which the instruction patterns comprise a pattern describing instruction semantics, in fig. 5. numeral 460. (Office Action, page 5)

Applicant's claim 13 recites "[t]he method of claim 1 in which the instruction patterns comprise a pattern describing instruction semantics."

As explained above, applicant's method matches the formatted memory access instructions to instructions patterns, and determines if two or more memory access instructions

can be supplanted by a single memory access instruction. As further explained in the originally filed application:

> **With memory access partitions such as a partition 200, 202, 204 it is convenient to perform memory access vectorization using instruction pattern matching. For each memory access instruction and instruction variance that is permitted in the source code language's definitions, an instruction pattern can be generated to describe the instruction semantics. The compiler 18 includes a store of all the instruction patterns. For example, as shown in FIG. 5, an instruction pattern 300 describes the SRAM memory read with a vectorization degree of 2, i.e., one that reads 2*MDAU bytes from SRAM. (page 6, lines 6-15 of the originally filed application)**

Thus, in performing the instruction pattern matching operations, the instruction patterns correspond to the semantics, or syntax, of the formatted instructions that are being matched.

As previously explained, Pentkovski, on the other hand, does not perform any type of matching operations, and particularly not one where instructions are being matched to instruction patterns that describe instruction semantics. Applicant also notes that reference numeral 460 to which the examiner referred in his rejection of claim 13 corresponds to the intermediate store operations performed using the intermediate buffer 362. As described in Pentkovski, "FIG. 5 is a diagram illustrating details of the intermediate stores 460 of FIG. 4, according to one embodiment of the invention" (col. 6, lines 17-19). Element 460, however, has nothing to do with matching instructions to instruction patterns and certainly not instruction patterns that describe instruction semantics. Accordingly, Pentkovski does not disclose or suggest at least "the instruction patterns comprise a pattern describing instruction semantics," as required by applicant's claim 13. Claim 13 is therefore patentable over the cited art.

Claim 25 recites "[t]he compilation method of claim 15 in which the instruction patterns comprises instruction semantics." For at least similar reasons as those provided with respect to claim 13, at least this feature is not disclosed by the cited art. Accordingly, claim 25 is patentable over the cited art.

Claim 26 depends from claim 25 and is therefore patentable for at least the same reasons as claim 25.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

No fee is believed due. Please apply any other charges or credits to deposit account 06-1050, referencing attorney docket 10559-886001.

Respectfully submitted,

Date: March 23, 2006

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21292017.doc